IBM 486 DX4 Microprocessor BIOS Writer's Guide

Application Note

Introduction

This document is intended for IBM 486 DX4 system BIOS writers. It is not a stand alone document but supplements other IBM 486 DX4 microprocessor documentation, including the **BLUE LIGHTNING 486DX2 DATABOOK** and **IBM 486DX4 DATABOOK Addendum**. This document includes information on IBM 486 DX4 microprocessor detection, 486 DX4 microprocessor configuration register definition, and recommendations for configuration register programming.

Configuration Register Index Assignments

The IBM 486DX4 processor provides on-chip configuration registers used to control the on-chip cache, system management mode (SMM) and other IBM 486 DX4 microprocessor unique features. Access to the configuration registers is achieved by writing the index of the register to I/O port 22h. I/O port 23h is then used for data transfer. Each I/O port 23h data transfer must be preceded by an I/O port 22h register index selection, otherwise the second and later I/O port 23h operations are directed off-chip and produce external I/O cycles. Reads of I/O port 22h are always directed off-chip. Table 1 lists the IBM 486 DX4 microprocessor configuration register index assignments.

If the register index number is outside the C0-CFh or FE-FFh ranges external I/O bus cycles occur. The configuration registers are described in more detail later in this document. Appendix A contains example code for accessing the IBM 486 DX4 microprocessor configuration registers.

Register Index	Register Name	Acronym	Width in Bits
00h-C0h	Reserved	--	--
C1h	Configuration Control 1	CCR1	8
C2h	Configuration Control 2	CCR2	8
C3h	Configuration Control 3	CCR3	8
C4h-CCh	Reserved	--	--
CDh, CEh, CFh	SMM Address Region	SMAR	24
D0h-FDh	Reserved	--	--

Register Index	Register Name	Acronym	Width in Bits
FEh	Device ID 0	DIR0	8
FFh	Device ID 1	DIR1	8

Table 1. Configuration Register Index Assignments

Detecting an IBM 486 DX4 Microprocessor

System BIOS can determine if an IBM 486 DX4 microprocessor exists by first determining if an IBM CPU exists. If an IBM CPU exists, the CPU's DIRs can be read to identify the type of CPU. Previous versions of IBM's CPUs did not contain the DIRs; however, all current CPU's contain DIRs.

Detecting an IBM CPU

Detection of an IBM CPU is done by checking the state of the undefined flags following execution of the divide instruction which divides 5 by 2. The undefined flags in an IBM microprocessor remain unchanged following the divide. Appendix B contains example code for detecting an IBM CPU.

Identifying an IBM CPU

Once it is determined that an IBM microprocessor exists, its DIRs can be read to identify its type. The DIRs contain CPU device identification, stepping and revision information. The DIRs are a subset of the IBM 486 DX4 microprocessor's configuration registers. The IBM 486 DX4 microprocessor's DIRs exist at register indexes FEh and FFh as shown in Table 1 and contain the information shown in Table 2.

Register	Description	Bit Position	Contents	Core/Bus Clock Ratio
DIR0	Device ID Reg 0	7-0	DEVID(7-0)= 1Bh DEVID(7-0)= 1Fh	2/1 (2x mode) 3/1 (3x mode)
DIR1	Device Stepping Device Revision	7-4 3-0	SID(3-0)=xxh RID(3-0)=xxh	

Table 2. IBM 486 DX4 Microprocessor Device Identification Register Contents

CPU EDX Value After RESET

Some CPU detection algorithms may use the value of the CPU's EDX register following RESET. The IBM 486 DX4 microprocessor's EDX register contents following reset are shown below.

```
        EDX[15:8] = DIR1(same contents as Device Identification Register 1)
        EDX[7:0]  = DIR0(same contents as Device Identification Register 0)
OR
        EDX[15:8] = 04h
        EDX[7:0]  = 90h
```

Configuration Register Bit Definitions

On-chip configuration registers are used to control the on-chip cache, system management mode and other IBM 486 DX4 microprocessor unique features. All bits in the configuration registers are initialized to zero following reset unless specified otherwise. The appropriate register settings will vary depending on system design. Therefore, the BIOS creating utilities or setup screens must have the capability to easily define and modify the contents of these registers. This will allow OEMs and integrators to easily configure these register settings with the values appropriate for the system design. The following paragraphs describe the categories of configuration registers, their purpose and bit assignments.

Configuration Control Registers CCR(1-3)

There are four registers in the IBM 486 DX4 microprocessor that control the cache, power management and other unique features. Figures 1 through 3 and Tables 3 through 5 describe the CCRs and briefly describe their applications.

Figure 1. Configuration Control Register 1 (CCR1)

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Reserved	*Reserved*	*Reserved*	NO_LOCK	MMAC	SMAC	SMI	RPL

Table 3. CCR1 Bit Definitions

Bit Name	Bit No.	Description
RPL	0	**Enable RPL Pins** If set, enable output pins RPLSET(1-0) and RPLVAL#. If clear output pins RPLSET(1-0) and RPLVAL# float.
SMI	1	**Enable SMM Pins** If set, SMI# and SMADS# pins are enabled. If clear, SMI# pin is ignored and SMADS# pin floats.
SMAC	2	**System Memory Memory Access** If set, any access to addresses within the SMM address space access system management memory instead of main memory. SMI# input is ignored while SMAC is set. Setting SMAC=1 allows access to SMM memory without entering SMM. This is useful for initializing or testing SMM memory.
MMAC	3	**Main Memory Access** If set, data accesses to addresses within the SMM address space are issued to main memory instead of system management memory. This is only used within an SMM service routine to access normal memory which overlaps SMM memory.
NO_LOCK	4	**Negate LOCK#** If set, locked cycles are inhibited for instructions that are considered as locked instructions by the CPU. These instructions include interrupt acknowledge cycles, descriptor loads, and updates and accesses to the interrupt descriptor table. However, locked cycles are not inhibited by No_Lock bit for TLB table look ups, XCHNG instructions to memory, and any instruction that includes a lock prefix. If clear (No_Lock=0), locked cycles occur for instructions that are considered as locked instructions by the CPU.
Reserved	7-5	

Figure 2. Configuration Control Register 2 (CCR2)

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
SUSP	BWRT	BARB	WT1	HALT	LOCK_NW	WBAK	*Reserved*

Table 4. CCR2 Bit Definitions

Bit NAME	Bit No.	Description
Reserved	0	
WBAK	1	**Enable Write-Back Cache Interface Pins** If set, enables the CPU write-back cache interface pins which include CACHE#, INVAL, WM_RST, and HITM#. If WBAK is set to 0, INVAL and WM_RST are ignored and the HITM# and CACHE# outputs float.
LOCK_NW	2	**LOCK NW Bit** If set, the NW bit in CR0 becomes read only, and the CPU ignores any writes to this bit. This should be set to 1 after setting the CR0 NW to prevent inadvertent modification of the NW bit.
HALT	3	**Suspend on Halt** If set, execution of the HALT instruction causes the CPU to enter low power suspend mode. This bit should be used cautiously since the CPU must recognize and service an INTR, NMI, SMI or RESET to exit the "HALT initiated" suspend mode.
WT1	4	**Write-Through Region 1** If set, designates that any cacheable accesses in the 640 KBytes to 1 MByte address region are defined write-through. With WT1=1, any write to a cached value in this range will also get issued to the external bus.
BARB	5	**Enable Cache Coherency on Bus Arbitration** If set, enable write-back of all dirty cache data when HOLD is requested and prior to asserting HLDA.
BWRT	6	**Enable Burst Write Cycles** If set, enables burst write cycles. This should only be set if the system logic supports burst writes. If set, the CPU will perform a 4 dword (16-byte) burst write on line replacements and write-back cycles as a result of snoop hits.
USE_SUSP	7	**Enable Suspend Pins** If set, SUSP# input and SUSPA# output pins are enabled. If clear, SUSP# input pin is ignored and SUSPA# output pin floats. These pins should only be enabled if the external system logic (chipset) supports them.

Figure 3. Configuration Control Register 3 (CCR3)

Bit 7	**Bit 6**	**Bit 5**	**Bit 4**	**Bit 3**	**Bit 2**	**Bit 1**	**Bit 0**
Reserved	*Reserved*	*Reserved*	*Reserved*	SMM_Mode	*Reserved*	NMI_EN	SMI_LOCK

Table 5. CCR3 Bit Definitions

Bit Name	**Bit No.**	**Description**
SMI_LOCK	0	**SMM Register Lock** If set, the CPU prevents modification of the following SMM configuration bits, except when operating in SMM: CCR1 USE_SMI,SMAC,MMAC,SM3 CCR3 NMI_EN SMAR Starting address and block size. Once set, the SMI_LOCK bit can only be cleared by asserting the RESET pin.
NMI_EN	1	**Non-Maskable Interrupt enable** If set, NMI interrupt is recognized while in SMM. This bit should only be set while in SMM, after the appropriate NMI interrupt service routine has been setup.
Reserved	2	
SMM_Mode	3	**SMM interface mode** 0=IBM Mode. Default is 0. 1=SL-compatible mode. If the SMI_Lock bit is clear, SMM_Mode may be modified. If the SMI_Lock bit is set, the SMM_Mode bit can no longer be modified. Once the SMI_Lock bit is set, the CPU must be reset in order to modify SMI_Lock and SMM_Mode. (reference DX4 Addendum, pages 9-11, for more details on SMM mode)
Reserved	4_7	

SMM Address Region Register - SMAR

The SMAR is used to define the SMM memory region. The SMAR has three 8-bit registers associated with it which define the region starting address and block size. Table 6 below shows the format for the SMAR and lists the index assignments for the SMAR starting address and block size. The region starting address is defined by the upper 12 bits of the physical address. The region size is defined by BSIZE(3:0) [bits 3:0 of SMAR). The BIOS and its utilities should allow for definition of the SMAR. There is one restriction when defining the SMM address region. The region starting address must be on a block size boundary. For example, a 128KByte block is allowed to have a starting address of 0K, 128K, 256K, and so on.

Table 6. SMAR Index assignments

Starting Address			**Region Block Size**
A31 - A24 Bits (7-0)	A23 - A16 Bits (7-0)	A15 - A12 Bits (7-4)	BSIZE (3-0) Bits (3-0)
CDh	CEh	CFh	

Table 7. BSIZE(3-0) Bit Definitions

BSIZE (3-0)	**Region Size**
0h	Disabled
1h	4 KBytes
2h	8 KBytes
3h	16 KBytes
4h	32 KBytes
5h	64 KBytes
6h	128 KBytes
7h	256 KBytes
8h	512 KBytes
9h	1 MBytes
Ah	2 MBytes
Bh	4 MBytes
Ch	8 MBytes
Dh	16 MBytes
Eh	32 MBytes
Fh	4 KBytes (same as 1h)

Programming Model Differences vs i486™

INVD and WBINVD Instructions

The INVD and WBINVD instructions are used to invalidate the contents of the internal and external caches. The WBINVD instruction first writes back any modified lines in the cache and then invalidates the contents. It ensures that cache coherency with system memory will be maintained regardless of the cache operating mode, write-through or write-back. Following invalidation of the internal cache, the CPU generates special bus cycles to indicate that external caches should also write back modified data and invalidate their contents.

On the IBM 486 DX4 microprocessor, the INVD instruction functions identically to the WBINVD instruction. The IBM 486 DX4 microprocessor always writes all modified internal cache data to external memory prior to invalidating the internal cache contents.

Control Register 0 (CR0) CD and NW Bits

The CPU's CR0 register contains, among other things, the CD and NW bits which are used to control the on-chip cache. CR0, like the other system level registers, is only accessible to programs running at privilege 0, the highest privilege level. Table 6 lists the cache operating modes for the possible states of the CD and NW bits.

The CD and NW bits are set to one (cache disabled) after reset. For highest performance the cache should be enabled in write-back mode by setting the CD=0 and NW=1. Sample code for enabling the cache is in Appendix C. To completely disable the cache, it is recommended that CD and NW be set to 1. On the IBM 486 DX4 microprocessor, the cache can be disabled by only setting CD=1. The IBM 486 DX4 microprocessor cache will always accept invalidation cycles even when the cache is disabled.

Table 8. Cache Operating Modes

CD	NW	Operating Modes
1	1	Cache disabled
1	0	Cache disabled
0	1	Cache enabled in Write-back mode
0	0	Cache enabled in Write-through mode

Initialization Sequence

The IBM 486 DX4 microprocessor features and capabilities described above should be enabled/initialized during BIOS POST. Below is a recommended sequence for initializing the IBM 486 DX4 microprocessor.

1. Program WT1.
2. Define SMAR.
3. Program USE_SUSP
4. Program SUSP_HLT
5. Enable cache (see Appendix C for example code).
6. Program LOCK_NW.
7. Initialize/enable SMM (see **BL486DX2 DATABOOK**, Appendix A **SMM Programmer's Guide ,** and the **IBM 486 DX4 Microprocessor Addendum**)

Appendix A - Programming IBM 486 DX4 Microprocessor Configuration Registers

Sample code for setting USE_SMI=1 in CCR1

```
mov   al, 0c1h          ; set index for CCR1
out   22h, al           ; select CCR1 register
in    al, 23h           ; read current CCR1 value
or    al, 02h           ; set USE_SMI bit
mov   ah, al            ; save set bit data in ah
mov   al, 0c1h          ; set index for CCR1
out   22h, al           ; select CCR1 register
mov   al,ah             ; get back saved data to al
out   23h, al           ; write new value to CCR1
```

Appendix B - Detecting An IBM 486 DX4 Microprocessor

```
assume              cs:_TEXT
public      _ism1sc
_TEXT       segment     byte    public 'CODE'
;*********************************
;       Function:  int isibm ()
;       Purpose:        Determine if IBM 486DX4 CPU is present
;       Technique:      IBM 486DX4 CPUs do not change flags where flags
;                       change in an undefined manner on other CPUs
;       Inputs:                 none
;       Output:         ax == 1 if IBM 486DX4 present, 0 if not
;**********************************
_ism1sc     proc    near
            .486
            xor     ax, ax          ; clear ax
            sahf                    ; clear flags, bit 1 always=1 in flags
            mov     ax, 5
            mov     bx, 2
            div     bl              ; operation that doesn't change flags on IBM 486DX4
parts
            lahf                    ; get flags
            cmp     ah, 2           ; check for change in flags
            jne     not_m1sc        ; if flags changed, it is not IBM 486DX4
            mov     ax, 1           ; true IBM 486DX4 CPU
            jmp     done
not_m1sc:
            mov ax, 0               ; non-IBM 486DX4 CPU
done:
            ret
_ism1sc     endp
_TEXT ends
end
```

Appendix C - Enabling The IBM 486 DX4 MicroprocessorCache

Sample code for enabling the IBM 486DX4 write-back cache and setting LOCK-NW=1.

```
;****  disable cache
        mov     eax, cr0
        or      eax, 040000000h          ; CD bit in CR0 = 1
        mov     cr0, eax
;**** write back and invalidate the cache
wbinvd
;**** enable write back mode
        mov     eax, cr0
        and     eax, 0fffffffh   ; NW bit in CR0 = 1
        mov     cr0, eax
;**** set LOCK_NW=1 to prevent modification of NW
        mov     al, 0c2h                 ; set index for CCR2
        out     22h, al                  ; select CCR2 register
        in      al, 23h                  ; read current CCR2 value
        or      al, 04h                  ; set LOCK_NW=1
        mov     ah,al                    ; save or'd data in ah
        mov     al, 0c2h                 ; set index for CCR2
        out     22h, al                  ; select CCR2 register
        mov     al,ah                    ; get back saved data into al
        out     23h, al                  ; write new value to CCR2
;****  enable cache
        mov     eax, cr0
        and     eax, 0bfffffffh          ; CD bit in CR0 = 0
        mov     cr0, eax
```